OIPE JC11
JUL 06 2004
PATENT & TRADEMARK OFFICE

SUBSTITUTE SPECIFICATION (CLEAN VERSION)

A PLUGGABLE SERVICE DELIVERY PLATFORM

RECEIVED
JUL 0 9 2004
Technology Center 2100

BACKGROUND

1. Technical Field

This invention relates to a service delivery platform that supports many devices to access many services and more particularly, to a pluggable service delivery platform in which many devices and many services are pluggable.

2. Description of Related Art

Nowadays there are a lot of pervasive computing devices such as handheld PCs, smartphone, mobile phone, screenphone, pager, fax machine, etc. They all have computing power and people wish to use these existing devices to access the network and do e-business. But there also exists challenges, since current network infrastructure is designed for PC. At the same time different services have different features. Under this environment, some effort is needed to connect a new device to the network, if the backend system is changed, e.g., some new service is added, the application on the device must be changed (or added); similar effort is needed to changed the logic of backend services when new

devices roll out. With the rapid development of network computing, there is a need for a pluggable service delivery platform that can support many devices and many services in a flexible and scalable way.

Some companies have designed some platforms for supporting many devices to access many services, but these platforms are designed specifically for some devices and services and there is no flexible way to plug a new kind of device or a new type of service into the platform.

SUMMARY OF THE INVENTION

The pluggable platform of the present invention can be used to overcome the shortcomings of existing platforms. The platform can be used in e-business and support many kinds of devices to access many types of services, while at the same time, new devices or new services can be easily added to the platform. This is where "pluggable" comes from.

The pluggable service delivery platform of the present invention comprises:

- Device Abstraction Layer (DAL)

It accepts the requests from devices and transforms them into XML and then sends them to the kernel of the platform. It also transforms the response XML documents

from the platform into a device specific format for presentation. It further comprises:

(1) Common transcoding component, shared by all devices and used for transforming between different kinds of data formats.

(2) Device dependent component, transforming between (whatever kind of data format/tranporting protocol for that particular device) and (XML over HTTP).

- Service Abstraction Layer (SAL)

It abstracts the common requirements from different services as a service profile. For each kind of service in some domain, a wrapper (adapter) is provided. The wrapper is used for transforming between (legacy data format/network protocol) and (XML/HTTP).

- Kernel Service Engine

The functions provided by the platform kernel service engine include:

- manage profiles of users, devices and services (i.e., user/device/service)

- synchronize/asynchronize service engine

- interface with other platform components

- transfer information between components within the platform using XML.

The platform is "pluggable" in three aspects.

Firstly, it is pluggable in the sense that a new kind of device can be easily plugged into the platform so long as the device profile that describes the device capability is provided.

Secondly, it is pluggable in the sense that a new type of service can be easily plugged into the platform so long as the service profile that describes the service feature is provided.

Thirdly, it is pluggable in the sense that the components that constitute the platform are pluggable, so any one of the components can be replaced by third party products, so long as the latter comply with a predefined interface, such as a Java Servlet and LDAP.

The advantages of the platform of this invention over existing platforms are listed in Table 1.

A local network company had the similar idea of a many-to-many platform called LISP/6A. The table below shows the differences and why the platform of the present invention is superior.

TABLE 1

Features	LISP/6A	This platform
Transcoding	No. Format has to be customized for each kind of device	Yes. The same content can be adapted to different kind of PvC devices
Scalability	No	Yes. With Network Dispatcher & Web Traffic Express & Distributed Application Tester
Contents representation in platform	Fixed segments with annotation	Represented in XML, easy to be extended
Support synchronized and asynchronized communication	No	Yes
Flexibility for plug in new device	No	Yes
Flexibility for plug in new service	No	Yes
Componentized parts within platform	No	Yes

These and other aspects, features and advantages of the present invention will be described or become apparent by the following detailed description of the preferred embodiments, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a preferred embodiment of a pluggable service delivery platform, which focuses on components of a platform kernel.

FIG. 2 shows the servlet (service engine) running on WebSphere (an IBM Web application server).

FIG. 3 shows the data flow between the service engine and the backend service (such as stock service).

FIG. 4 shows the device abstraction layer (device-platform interface) of the pluggable service delivery platform of Fig. 1.

FIG. 5 shows the service abstraction layer (platform-service interface) of the pluggable service delivery platform of Fig. 1.

FIG. 6 shows an implementation of the present invention using a WAP phone to access services through the platform.

FIG. 7 shows the process of adding a new kind of device to the platform.

FIG. 8 shows the process of adding a new type of service to the platform.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

It is to be understood that the exemplary system modules and method steps described herein may be implemented in various forms of hardware, software, firmware, special purpose processors, or a combination thereof. Preferably, the present invention is implemented in software as an application program tangibly embodied on one or more program storage devices. The application program may be executed by any machine, device or platform comprising suitable architecture. It is to be further understood that, because some of the constituent system modules and method steps depicted in the accompanying Figures are preferably implemented in software, the actual connections between the system components (or the process steps) may differ depending upon the manner in which the present invention is programmed. Given the teachings herein, one of ordinary skill in the related art will be able to contemplate these and similar implementations or configurations of the present invention.

Before describing preferred embodiments in detail, the terms used in the present invention are first listed as follows:

TERMS USED:

1. XML (eXtensible Markup Language): as its name says, it is a kind of markup language endorsed by W3C (Web standard organization). It is used to describe/define structured data and separate the data with presentation (compared with HTML which strongly combines data and the presentation). The same set of data expressed by XML language can generate a different presentation (Markup Language) with a different style sheet language (XSL: eXtensible Stylesheet Language). Since data represented in XML are highly structured, they are very suitable for automatically exchanging information between applications. The adoption of XML for exchange between different components is the biggest characteristic of the invention.

2. WAP (Wireless Application Protocol): a wireless communication protocol specifically designed for handheld devices (especially mobile phones). WAP is critical for mobile phones to access the information on the net just as a PC using HTML/HTTP to access the Internet.

3. Servlet: Java small service application, a special kind of Java class running on a Web server. It can accept the request from the Web (browser), parse the parameters and execute the predefined logic (such data connection with backend system) and generate a response and send it

back to the browser. Since Java Servlets are written in Java they are cross platform (OS) and highly portable applications. Java servlet can dynamically generate different pages for different kinds of devices such as HTML for PC, WML for WAP phone, etc.

4. Trancoding: a kind of transformation technique which transforms the same set of data into different pages based on predefined criteria (such as display resolution, color depth, multimedia support). The technique further consists of many components, including an image transcoder (e.g. GIF -> JPEG, JPEG -> BMP, color -> grey -> black/white) and a text transcoder (e.g. text abstraction, text -> audio). Using a transcoding technique, one type of XML document can be transformed into another type of XML document and can further be transformed into some kind of presentation (e.g. HTML or WML) by style sheet language.

5. Device gateway: the device gateway in the invention sits in the device abstraction layer, it can accept a request from a device over some sort of network protocol, transform it into XML over HTTP, then send it to the platform kernel. After getting the data from the backend system through the platform kernel, it then transforms the page into a device readable page and sends it to the device over the network that the device connects to.

6. Service adapter (wrapper): the service adapter in the invention sits in the service abstraction layer, it transforms between the platform format (XML)/protocol(HTTP) and service specific format/protocol.

The following paragraphs will illustrate in detail how to implement the invention.

The pluggable service delivery platform shown in FIG. 1 comprises three parts, Device Abstraction Layer (DAL), Service Abstraction Layer (SAL) and Kernel Service Engine. FIG. 1 focuses on components of a platform kernel. The detail of SAL and DAL will be illustrated in FIG. 4 and FIG. 5 respectively. As shown in FIG. 1, the platform kernel comprises a service engine **101**, a runtime monitor **102**, a profile manager **103** and auxiliary components **104** (such as a billing manager **104a**, a security manager **104b**, etc.) As shown in FIG. 1, XML is used within the platform as an interface language. XML is used widely in the platform to exchange information between different components in the platform. XML is also used in the DAL and SAL, such that information processed in the platform will be based on XML. For the service engine, both a synchronized service engine and an asynchronized service engine are provided. For example, the synchronized service engine can be based on IBM

WebSphere which is a Web application server and has strong XML support.

FIG. 2 shows how servlets are organized in WebSphere. As shown, servlets are built on and managed by WebSphere (start, stop, add, delete). A chain of servlets can be called when a request from the device is sent to the platform, processed by the platform, till responded with a page. The first servlet called, which is also the most important one, corresponds to a called URL. The servlet can then call other servlets which form a servlet chain. As shown in FIG. 2, under WebSphere application server (Default Server), there is a ServletEngine which is the base Servlet engine. Under Servlet Engine there are many directories, such as "default_app", "admin", "examples", etc. Under some specific application, there have been some servlets that will be used. For example, under "default_app", there are "Snoop" servlet, "hello" servlet, "ErrorReporter" servlet, etc.

FIG. 3 shows the data chart and also the interaction between the service engine and the backend service (e.g. stock service).

The platform runtime monitor **102** is used to monitor the runtime status of platform.

The profile manager **103** is used to manage a user profile, device profile and service profile.

The user profile can include items as, user ID, user name, telephone number, etc.

The device profile can include items as, device ID, vendor name, device type, display resolution, multimedia capability, corresponding XSL (which is used to present XML data on that specific device).

The service profile can include items as, service ID, service provider, operating time, start URL, etc.

Besides the above components, the platform kernel can also include many auxiliary components, such as a device manager to manage device access, a service manager to manage service connection to the platform, an event manager to trigger some platform related event and send to user, a transaction manager, a billing manager, and a security manager. All the above components are pluggable and can be replaced by third party products.

In FIGS. 1-3, the kernel parts of the platform are described in detail. These components are used to manage user/device/service profile information, provide a synchronized/asynchronized service engine, use XML to exchange information and carry transaction related information between different parts of the platform. As shown in FIG. 1, the platform kernel consists of three layers: a runtime layer, an admin layer and a development layer. Platform APIs are used to interact between layers. The runtime layer provides online information access and

control, the management layer provides service such as add/delete user/device/service information, and the development layer provides support for a new device/service.

In FIG. 4 SAL will be described in detail.

There are mainly two portions in FIG. 4. The first portion, on top of the dotted line, is what we call Control mode. The administrator can use the UI under this mode to install new services and configure the existing services. Both the control mode portion and run-time mode portion have a PnP (plug and play) manager. The enumeration is used to abstract the common features of different services and differentiate them by different industrial. The second portion, under the dotted line, is what we call Run-time mode and has mainly three layers. The bottom layer is the enumeration. Specific industrial should have its specific drivers which should obey the open service protocol. The middle layer is the Service Abstract Layer (SAL). SAL abstracts the common requirement of different services. The upper layer is the kernel of run-rime mode and is called run-time unit. It further comprises several important parts. The first one and the most important one is the PnP manager that corresponds to the PnP manager in the Control mode portion. It has an event listener to listen to events coming from the service-platform interface. It manages

the plugged services to the platform. Then there's the maintenance manager. It's used to manage the lifetime of a service like when it's opened and closed, when it will expire, etc. The third one is the resource manager. Then there is the security manager to manage the security in the platform to securely transfer messaged and documents. The total service design architecture is EVENT DRIVEN and STATE BASED. The relationship between different layers is like this. The Service Abstract Layer enumerate (maintain) and administrate services and report events to the run-time unit. It also works with the run-time unit to manage transactions to make sure that several commands from one transaction will not be broken into pieces. The main event types include a New service event, an Update event, etc. All the events are related to service-platform interaction and platform operation.

In FIG. 5 DAL will be described in detail.

There are also two parts in FIG. 5. The one above the dashed line includes Profile Generating Tools to generate a profile for some new device. The information will be saved in the registry and can be accessed by the profile manager of the administrator UI. The other part under the dashed line consists of a Device Abstract Layer, a Profile Manager, and Run-time managers. The run-time managers then include a Protocol manager, a Connection manager, a Contents manager and an event

manager. A common interface (Device Abstract Layer) is needed to define the common behavior of PvC devices. The devices may connect to the platform in different ways (e.g. LAN/WAN, PSTN, GSM/CDMA and CDPD), so gateways are needed for each kind of connection. No matter how the device is connected to the platform, the devices' rich features can be extended from this common base. And this common base is expressed in XML too. The profile managers serve as the focal point between the platform device administrator and the platform run-time kernel. The features of the device are saved in the registry as (key, value) pairs. Protocol manager is used to decide whether to send the message through IP or HTTP protocol in the platform. Connection manager is used to manage the connect in a transaction, i.e. set up the connect when device request, or send the message when certain conditions meet. Contents manager is built upon the transcoding technique. It decides how to send out the message. It assembles the contents based on the devices' profile. Event manager generates system events when a device contacts the platform. (Certain profile header should be provided in the head of the message that the device sends). No matter how the device accesses the platform (through GSM, CDPD, PSTN, LAN or other ways), it is required to include a description of the device

(profile) in the header of the message it sends to the platform when it is logged on.

In the above paragraphs, a preferred embodiment of a pluggable platform according to the present invention is illustrated. The platform has the following advantages: no matter what kind of device the end users use they can always access the key information in a consistent and natural way and all the returned pages will fit well on that device. This also simplified the service connection process. For now, the service providers need only one reserved line to connect the service to the service delivery platform.

FIG. 6 shows how a service can be hosted on the platform. To be specific, the process of using WAP phone to access the services through the platform is shown. Firstly, various WAP phones connect to the WAP gateway through GSM network and then data channels (PSTN connection or ISDN connection). The information before WAP gateway is binary WML over WAP, while after the WAP gateway, it will be WML over HTTP. When a user uses the WAP phone, some URL has actually been selected, the request will be sent to a servlet that corresponds to the URL. The servlet will analyze the request parameter, call some service wrapper as required, then get data from background services. This kind of data connection is common to domain service and independent of the specific

service provider. After getting the data, the servlet will reorganize the data and generate a page (e.g. HTML or WML). The page can be generated by transcoding means after retrieving the device style sheet (XSL) stored in device profile through LDAP call.

FIG. 7 shows how to plug a new kind of device. When adding a new kind of device the system administrator can use the admin tools and select "Add New Device" item, then fill in a form to generate a device profile in the profile manager. Among the description, XSL is used to describe device capability. At the runtime, when the platform receives user requests, on one hand, it will generate XML data based on the return from service, and on the other hand, it will retrieve the device profile from the profile manager, then generate the final page layout based on the transcoding technique.

FIG. 8 shows how to plug a new kind of service. When adding a new kind of service the system administrator can use the admin tools and select "Add New Service" item, then fill in a form to generate a service profile in the profile manager. At the runtime, when the user connects to the platform, only a dynamic service list that the user subscribes will be listed.

Although illustrative embodiments of the present invention have been described herein with reference to the accompanying drawings, it is to be understood that

the present invention is not limited to those precise embodiments, and that various other changes and modifications may be affected therein by one skilled in the art without departing from the scope or spirit of the invention. All such changes and modifications are intended to be included within the scope of the invention as defined by the appended claims.